pLLM: Guidance on Nutritional Intake (GINI) Project Module

Swapnil Jha (swapnil.jha@uni-potsdam.de)

Cognitive Systems at University of Potsdam

Introduction

GINI is a nutrition-focused chatbot that processes user input through a GPT-3 pre-trained model, generating text-based responses tailored to its users.

Problem Objective

Gini is a personalized nutrition chatbot designed to offer three key features: customized meal planning, recipe recommendations, and comprehensive nutritional information. It adapts its responses to align with each user's unique dietary needs and preferences.

Problem Statement

Gini tackles common nutritional challenges, such as difficulties with meal planning, limited recipe choices, and a lack of detailed nutritional information. The platform not only helps users improve their eating habits but also educates them, promoting healthier decisions in their everyday lives.

Goals and Deliverables

The system's goals and deliverables were outlined as follows:

- Develop a minimum viable product (MVP) that includes personalized meal planning, recipe suggestions, and detailed nutritional analysis.
- Implement a feedback mechanism or question-chaining feature with memory to enhance and refine future recommendations.

System Overview

Architecture Diagram

The following figure depicts the system architecture, highlighting the interaction between the user interface, the LLM backend, and LangChain, which integrates the OpenAI API.

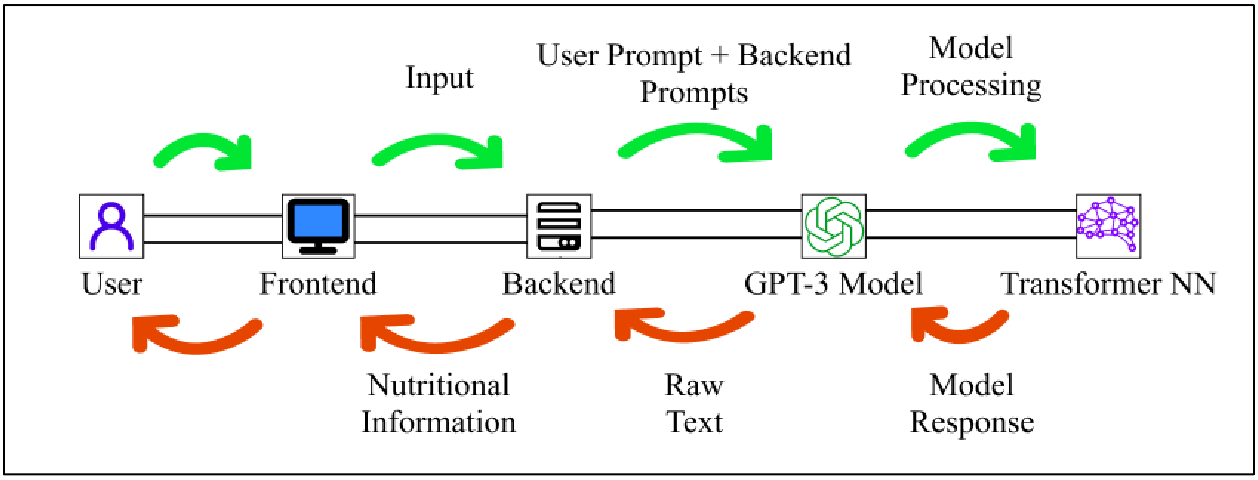


Figure 1: System Architecture

Technology Stack

The development of the system utilized the following technologies:

- **Frontend**: Streamlit, a Python library for building user interfaces.
- **Backend**: Python combined with LangChain, leveraging OpenAI's GPT-3.5 Turbo for natural language processing and response generation.
- **Deployment**: Streamlit Community Cloud for hosting the application.

Workflow Overview

The workflow is structured as follows:

- **User Input**: The user submits a query through a text box, indicating their preferences and dietary restrictions
- **Router Chain**: The system routes the input to the most suitable prompt based on its content, such as recipe suggestions or meal plans.
- **Prompt Processing**: The selected chain processes the query using the GPT-3.5 model along with the relevant prompt template.
- **Response Display**: The chatbot's output, which may include meal plans, recipes, or nutrient breakdowns, is presented on the Streamlit interface.
- **Further Queries**: Users can ask follow-up questions, and the system similarly handles these, ensuring a seamless interaction.

Tools and Libraries Used

1. Libraries:

LangChain:

- Utilized for constructing prompt chains and managing routing between various prompt templates.
- Facilitates the integration of the language model (LLM) with custom templates for generating dynamic responses.

- Key submodules employed include:
    - **MultiPromptChain**
    - **LLMRouterChain**
    - **RouterOutputParser**
    - **LLMChain**

Streamlit:

- Provides the user interface (UI) for the chatbot, enabling the creation of an interactive web-based frontend where users can submit queries, view responses, and engage with the system.
- Important Streamlit components used:
    - **st.text_area**: To capture user queries.
    - **st.button**: To process user input and trigger responses.
    - **st.markdown**: To display formatted chatbot responses.

OpenAI API (GPT-3.5-turbo):

a. The language model (LLM) used for generating personalized responses based on user input.

b. Integrated via the **ChatOpenAI** class from LangChain.

IPython Display:

Specifically employs Markdown from the IPython library for rendering Markdown content. While typically used in Jupyter notebooks, its application in this script is aimed at formatting output in Markdown.

Gini's Approaches and the Final Model being used

Gini underwent several iterations before the team concluded that utilizing a pre-trained model like GPT-3.5 Turbo would yield the best results. The following sections will outline the key factors that influenced these decisions throughout the project's development.

Initial Architecture of Gini: A "Bare-bones Approach"

Initially, Gini employed a fine-tuned Transformer-based model (BERT) trained on a custom dataset consisting of recipes, nutrients, and dietary restrictions. This model was designed to classify user queries, generate text outputs (such as meal plans), and provide structured information like nutritional data. However, the team encountered challenges related to the accuracy and coherence of the generated outputs, despite extensive testing of various hyperparameters.

Ultimately, due to resource limitations and subpar model performance, the team opted to abandon this approach. They chose instead to leverage a pre-trained model, which demonstrated significantly better results compared to the initial "bare-bones approach."

Current Approach Being Used by GINI

The current iteration of Gini leverages GPT-3.5 Turbo, providing an advanced method for generating highly personalized meal plans, recipe suggestions, and nutritional advice. Below are key insights into how Gini employs GPT-3.5 Turbo to accomplish its objectives and deliver coherent, persuasive results:

Leveraging GPT-3.5-turbo's Natural Language Understanding

Natural Language Proficiency: GPT-3.5 Turbo excels at understanding and generating natural language, making it an ideal choice for interactive user conversations. Gini leverages this capability to accurately interpret user inputs, ranging from straightforward questions like "What should I eat for dinner?" to more nuanced dietary needs such as "I need a high-protein, vegan meal plan."

Insight: The model's ability to parse complex sentences, identify key entities (such as ingredients or dietary restrictions), and generate contextually relevant responses enables Gini to provide personalized meal suggestions in a conversational manner that feels intuitive to users.

Multi-Prompt Chain System for Specialized Queries

Multi-Prompt Chain System: Gini employs a multi-prompt chain system through LangChain, where different types of queries—such as meal planning, recipe suggestions, and nutrient breakdowns—are directed to their corresponding prompt templates. This structured approach enables Gini to produce focused and relevant responses tailored to user inputs.

Insight: By categorizing user queries into specialized prompts, Gini enhances the relevance and accuracy of its outputs. For example, when a user requests a "balanced meal plan," the system routes the query to a specific prompt that generates a comprehensive daily or weekly plan. In contrast, if another user seeks a "nutrient breakdown," their query is directed to a prompt specifically designed to provide detailed information on macronutrients and micronutrients.

Prompt Engineering for Personalization

Tailored Prompt Templates: Gini utilizes meticulously crafted prompt templates for each of its core functionalities, including recipe suggestions, nutrient breakdowns, and dietary advice. These templates are designed to harness GPT-3.5 Turbo's capacity to adjust its language generation based on the provided instructions.

Insight: Gini enhances personalization by incorporating user-specific details directly into the prompts. For instance, if a user expresses a preference for "high-protein meals," the corresponding prompt will reflect this dietary restriction, enabling GPT-3.5 Turbo to generate suggestions that are closely aligned with the user's nutritional goals.

Sample Prompt:

"You are an expert nutritionist. Provide a meal plan for a person who follows a high-protein vegan diet. Include recipes and a detailed nutrient breakdown for each meal."

Real-Time Response Generation

Real-Time Response Generation: GPT-3.5 Turbo is a pre-trained model that enables real-time generation of responses based on user input. Gini leverages this capability to deliver instant feedback, whether it's creating a recipe or providing nutritional advice, without relying on predefined datasets or rigid rules.

Insight: By utilizing GPT-3.5 Turbo's generative abilities, Gini can offer dynamic, on-the-fly meal suggestions that feel both fresh and personalized. The model's capacity to generate coherent and logical steps—such as in recipe instructions—enhances the user experience, creating an interaction that is both conversational and informative.

Handling Ambiguity and Complex Queries

Handling Ambiguous Queries: One of GPT-3.5 Turbo's key strengths is its capacity to manage ambiguous or open-ended queries. Gini takes advantage of this feature by allowing users to submit general requests, prompting GPT-3.5 Turbo to respond with relevant options and often guiding users with clarifying follow-up questions (e.g., "Would you prefer a vegetarian or vegan meal?").

Insight: This flexibility enables Gini to effectively address vague queries such as "I want something healthy," returning contextually appropriate suggestions. The model's ability to interpret unclear inputs and provide sensible clarifications significantly enhances user satisfaction.

Context Awareness and Conversational Flow

Context Maintenance: GPT-3.5 Turbo can retain context throughout multiple turns of a conversation, which is crucial for Gini's interactive style. For instance, if a user requests a meal plan and subsequently provides specific dietary restrictions or preferences, the model can seamlessly incorporate that new information into the ongoing discussion.

Insight: This context awareness allows Gini to create a fluid conversational experience, where follow-up questions or adjustments to meal plans feel natural and cohesive with previous interactions. This enhances user engagement and satisfaction by making the dialogue more relevant and personalized.

Scalability and Adaptability

Scalability and Adaptability: As a pre-trained model with extensive knowledge across various domains, GPT-3.5 Turbo enables Gini to scale and adapt to new use cases without the need for retraining. This flexibility allows Gini to address a wide range of dietary needs and continuously enhance its feature set—such as incorporating fitness-based meal plans or accommodating more complex dietary restrictions—by simply updating the prompt templates.

Insight: This scalability empowers Gini to evolve without significant re-engineering efforts, making it highly adaptable to future demands. For instance, it could easily integrate APIs for real-time nutritional data or establish user feedback loops to refine its recommendations, ensuring that Gini remains relevant and effective as user needs change.

Ensuring Coherence and Persuasiveness

High-Quality Responses: The responses generated by GPT-3.5 Turbo consistently exhibit high quality in terms of coherence and logical flow, particularly when managing complex tasks such as suggesting recipes with detailed step-by-step instructions or providing a comprehensive nutrient breakdown for a full-day meal plan.

Insight: By leveraging GPT-3.5's ability to produce convincing and well-structured text, Gini offers users not only informative responses but also outputs that feel professionally curated. This enhances the system's trustworthiness and usability, particularly for users looking for reliable nutritional advice.

Future Expandability with External APIs

F**uture Scalability**: Although Gini currently depends on GPT-3.5 Turbo, the system has been architected with future scalability in mind. The planned integration of external APIs for live nutritional data, such as USDA FoodData Central, will enable Gini to access real-time nutritional updates, ensuring that meal plans and advice are consistently current.

Insight: While GPT-3.5 Turbo serves as a robust foundation for general language generation tasks, the forthcoming integration of external APIs will significantly improve Gini's accuracy and relevance, particularly in instances where nutritional data is frequently updated. This enhancement will ensure that users receive the most accurate and up-to-date information possible.

Core Components of GINI

Recipe Suggestions

Gini provides tailored and nutritious meal ideas, utilizing the **RecipeSuggestions_template**, to create recipes tailored to the user's preferences and dietary needs. Users can input their available ingredients, and the system will generate recipes with detailed instructions, cooking times, and nutritional information. This ensures that the suggested meals cater to various dietary preferences, such as vegetarian, gluten-free, or protein-rich options. Gini's recipe suggestions are designed to offer balanced nutrition, going beyond simple meal ideas.

Here are the main steps:

1) Users provide a list of ingredients they have on hand.
2) The router chain selects the **RecipeSuggestions_template** to guide the recipe generation process.
3) The OpenAI model creates a personalized list of recipes, each featuring cooking instructions, prep time, and a comprehensive nutrient analysis, including calories, proteins, fats, carbs, vitamins, and minerals.

Nutrient Breakdown

With Gini's **NutrientBreakdown_template**, users can access comprehensive nutritional information about various food items. This feature allows users to input a specific food item and receive a detailed analysis of its nutritional content, making it an invaluable tool for informed dietary choices. The system provides a breakdown of macronutrients, including proteins, fats, and carbohydrates, as well as micronutrients like vitamins and minerals. This ensures that users can understand the nutritional value of their food, whether it's a single ingredient or a complex meal.

Here's a summary of the process:

1) Users initiate the process by entering a food item they want to analyze.
2) The **NutrientBreakdown_template** activates the LLM to generate a detailed nutritional profile.
3) The output includes essential nutritional facts, such as calorie count, macronutrient distribution (protein, fats, and carbohydrates), and micronutrient content (vitamins and minerals).

Meal Plan generation

Gini crafts tailored meal plans based on users' dietary preferences and objectives (like weight loss, muscle growth, or balanced eating). The **MealPlan_template** dynamically creates plans spanning a day or week, adapting to the user's request. Each meal comes with a comprehensive nutrient analysis, ensuring it aligns with the user's health targets. Gini incorporates user input preferences (e.g., vegetarian, high protein) to deliver balanced, customized meal plans that support specific dietary goals.

Key points:

1) Users input their dietary preferences and goals (e.g., weight loss, muscle gain).
2) The **MealPlan_template** is launched, employing an LLM to generate a personalized meal plan.
3) The system offers a nutrient breakdown for each meal, ensuring users meet their caloric and macronutrient needs.

Exercise And Nutrition Alignment

This feature synchronizes nutrition with the user's workouts, offering detailed pre- and post-workout meal advice. Leveraging the **ExerciseNutrition_template**, Gini crafts meals based on the user's exercise type, duration, and intensity. It considers nutrient timing, like boosting carbohydrate intake before workouts for energy and protein after workouts for recovery. This integration enables Gini to offer targeted nutrition guidance that supports fitness goals.

Key points:

1. Users input their exercise routine details, such as workout duration and intensity.
2. The **ExerciseNutrition_template** analyzes this input and generates personalized meal recommendations.
3. Nutrient timing suggestions are provided to ensure peak energy and recovery.

Dietary Restrictions Handling

Gini caters to dietary restrictions by offering safe food choices and recipes tailored to specific needs, like gluten intolerance, veganism, or dairy allergies. The **DietaryRestrictions_template** ensures that suggested meals and recipes comply with the user's restrictions while maintaining balanced nutrition. The system also recommends substitutions for common allergens and offers creative alternatives to meet dietary requirements without compromising nutritional value.

Key points:

1) Users input their dietary restrictions (e.g., gluten-free, vegan).
2) The **DietaryRestrictions_template** is selected and processes the input.
3) Gini generates a list of allowed foods, avoids restricted items, and provides recipes that adhere to the restrictions.

Weight Management

Gini helps users manage their weight through personalized meal plans and expert tips for weight loss, gain, or maintenance. The **WeightManagement_template** provides tailored advice on caloric intake, portion control, and nutrient distribution. Whether aiming to lose fat or build muscle, Gini ensures that the recommended meal plans align with the user's specific caloric and macronutrient needs.

Key points:

1) Users specify their weight management goal (loss, gain, or maintenance).
2) The **WeightManagement_template** processes this goal and generates meal plans and tips.
3) The system includes guidance on portion control, creating a caloric deficit or surplus, and nutrient distribution to help users achieve their desired weight.

Food Allergy Management

Gini helps users manage food allergies by identifying common allergens and recommending safe alternatives. Using the **FoodAllergies_template**, the system provides users with a list of foods to avoid, based on their specific allergies, and suggests safe recipes that eliminate these allergens. This component ensures that users receive safe, allergy-free meal recommendations without compromising on nutrition.

Key points:

1) User inputs their specific food allergies.
2) The **FoodAllergies_template** triggers the system to identify allergens and safe alternatives.
3) Gini generates allergy-friendly recipes and suggests safe foods while avoiding common allergens.

Digestive Health Guidance

Gini provides recommendations for maintaining digestive health using the **DigestiveHealth_template**. The system suggests foods that promote gut health, such as high-fibre options, and offers advice on managing common digestive issues like bloating or indigestion. Users receive practical tips on habits and foods that can improve digestion, helping them maintain a healthy gut and overall well-being.

Key points:

I. User inputs concerns related to digestive health.
II. The **DigestiveHealth_template** processes the input and generates advice on improving digestion.
III. Gini suggests gut-friendly foods, habits, and tips to alleviate digestive discomfort.

Plant-based Diet Advice

Gini provides thorough guidance for users following a plant-based diet through the **PlantBasedDiet_template**. The system ensures that users receive sufficient protein, vitamins, and minerals from plant sources and offers balanced meal suggestions that align with their dietary preferences. Gini's meal plans help users achieve nutritional balance while maintaining a plant-based lifestyle.

Key points:

1. Users input their preferences for a plant-based diet.
2. The PlantBasedDiet_template generates balanced meal suggestions that emphasize plant-based protein and micronutrient intake.
3. Gini offers meal plans that ensure users meet their nutritional needs while adhering to a plant-based lifestyle.

Implementation Details

Template Design for Prompts

Gini employs a modular approach to handle user queries through custom-designed prompt templates. Each template is meticulously crafted to generate contextually accurate responses for various categories of queries, including recipe suggestions, nutrient breakdowns, meal plans, and dietary advice. These templates serve as structured instructions for the language model (GPT-3.5-turbo) to interpret user inputs and respond effectively.

Key points:

1. Each prompt is embedded in a chain, with variable placeholders (e.g., {input}) that are dynamically filled based on user input.
2. For example, the **RecipeSuggestions_template** takes user-provided ingredients and generates a list of recipes along with nutrient breakdowns, while the **MealPlan_template** processes dietary preferences and produces a full meal plan.
3. These prompt templates ensure that Gini can handle a wide variety of queries, ranging from simple nutrient requests to more complex meal-planning tasks.

Multi-Prompt Chain Setup

The multi-prompt chain system is integral to Gini's functionality, enabling dynamic routing of user queries to the most appropriate prompt template. This architecture ensures that the system responds accurately based on the type of query, without overwhelming the language model with irrelevant details.

Key points:

1. The *MultiPromptChain* is implemented by creating individual chains for each prompt template (e.g., **NutrientBreakdown_template**, **RecipeSuggestions_template**).

2. A router chain (*LLMRouterChain*) is configured to determine which prompt chain to activate based on the user's input.

3. The decision-making process is further refined by a *RouterOutputParser*, which modifies or adjusts the input if necessary before it is passed to the corresponding prompt template.
4. This setup reduces errors and increases the specificity of responses, as each query is funnelled through a specialized template, ensuring accurate, context-aware output.

Handling User Input

The system's approach to handling user input is both structured and dynamic, ensuring that users receive accurate and tailored responses. It captures inputs such as dietary preferences, goals (e.g., muscle gain, weight loss), or dietary restrictions, and routes them to the appropriate prompt via the router chain.

Key points:

1. Inputs are captured through Streamlit's text area widgets, where users can describe their dietary preferences, ingredient lists, or health goals.
2. The input is then processed by the multi-prompt chain system, where the router determines the type of query (e.g., recipe suggestion, nutrient breakdown).
3. The system dynamically fills the placeholder variables in the prompt templates (e.g., {input}) with user-provided data and passes it to the LLM for generating the final response.
4. The input handling mechanism is flexible, allowing users to adjust queries based on feedback, which triggers the appropriate response modification.

Data Flow and Decision Routing

The decision-routing system is crucial for ensuring that each query is processed through the most relevant prompt chain, providing personalized and contextually relevant outputs.

Key points:

1. When a user submits a query, it is first evaluated by the Router Chain. This chain determines which prompt (e.g., ExerciseNutrition_template, MealPlan_template) is best suited for processing the input.
2. The router uses a predefined set of candidate prompts, listed as destinations, which are dynamically referenced in the multi-prompt chain.
3. Based on the decision made by the router, the system routes the input to the correct chain, which then engages the GPT-3.5 model to generate the desired output.
4. The data flow ensures efficient processing of inputs, avoids irrelevant prompts, and improves response accuracy through targeted routing.

Error Handling and Edge Cases

Error handling is essential to ensuring that Gini provides accurate responses even when faced with incomplete, ambiguous, or out-of-scope user inputs. The system includes mechanisms for managing such edge cases, prompting users for clarification or defaulting to a generic response when necessary.

Key points:

1. **Input Guard for Relevance**: The system incorporates an input guard to check if the prompt is relevant to nutrition or health. If the input is invalid, the system responds with "INVALID PROMPT," ensuring that irrelevant queries do not result in incorrect or confusing outputs.
2. **Handling Incomplete Inputs**: In the case of incomplete inputs (e.g., missing ingredient details for a recipe suggestion), the system can prompt the user for more information or default to a basic response based on the available data.
3. **Refining Vague Inputs**: Vague inputs are handled by the router chain's ability to adjust or revise the input before routing it to the LLM. For example, if the user's query lacks context, the system will refine the query to better suit the selected prompt.
4. **Feedback Loop and Clarification Mechanism**: By incorporating this feedback loop and clarification mechanism, the system ensures that users receive relevant, high-quality responses even when faced with ambiguous queries.

Evaluation and Testing

Functional Testing of Components

Functional testing was carried out to ensure that Gini's core components—**meal planning**, **recipe suggestions**, and **nutrient breakdown**—performed correctly and met the system's functional requirements. The aim was to verify that the system could handle various input scenarios and generate relevant outputs that aligned with the user's dietary goals.

Performance Evaluation

Performance testing focused on ensuring that Gini responded to user queries within an acceptable timeframe and could handle multiple simultaneous requests without a significant degradation in performance.

Technical Breakdown:

I. **Response Time**: Tests were conducted to measure Gini's response time under different load conditions (e.g., multiple concurrent users, complex meal plan generation). The system's backend was designed to handle varying levels of user traffic while maintaining responsive interactions.

II. **Scalability**: The system's backend, deployed on cloud platforms (e.g., **Render.com**), was evaluated for its ability to scale effectively. Load testing was performed to ensure that Gini could handle an increased number of users without crashing or slowing down significantly.

Accuracy of Nutrient Data

To ensure that Gini's nutrient breakdown feature provided reliable and accurate information, the nutrient data generated by the system was compared against known nutritional databases such as **USDA FoodData Central**.

Technical Breakdown:

I. **Data Validation**: The nutritional data (e.g., calorie count, macronutrient content) generated for various foods and recipes were cross-referenced with reliable databases and resources to verify accuracy. This helped ensure that users could trust the information provided by Gini.

II. **Continuous Improvement**: As part of the development roadmap, future iterations of Gini will include API integrations with external databases like USDA to further enhance the accuracy and comprehensiveness of the nutrient data.

Challenges Faced

Technical Challenges

One of the primary technical challenges was **integrating the LangChain model** with the overall system architecture while ensuring compatibility across different platforms (e.g., Streamlit for the front end and LangChain for backend prompt handling). This required careful management of API calls and decision-routing within the multi-prompt chain system to ensure that each user input was processed correctly and led to a meaningful response.

Another significant challenge was handling **vague or complex user inputs**. Users often submitted queries that lacked specificity (e.g., "I want a healthy meal"), which made it difficult for the system to generate personalized and accurate responses. To address this, the prompt templates had to be refined, and additional clarification mechanisms were implemented to prompt users for more specific information. This process required ongoing adjustments to improve the model's ability to interpret and respond to diverse input types.

Key Technical Challenges:

I. Integrating LangChain with the user interface and ensuring smooth backend interactions.

II. Managing and refining prompt templates to handle ambiguous or complex user inputs.

Time Constraints

The development of Gini was impacted by **time constraints**, which limited the implementation of more advanced features. While the system successfully developed a functional MVP with key features like meal planning, recipe suggestions, and nutrient breakdowns, certain enhancements, such as **long-term meal tracking** and **feedback-based learning**, were deprioritized due to the tight project timeline.

Key features that were deprioritized:

I. **Long-term meal tracking**: A feature that would allow users to monitor their dietary habits over weeks or months for better health management.

II. **Feedback-based learning**: The system's ability to adapt to user preferences over time by learning from interactions was delayed in favour of delivering core functionality.

Handling Complex Dietary Restrictions

Managing **complex dietary restrictions** was a major challenge for Gini. Users with multiple, overlapping restrictions (e.g., vegan, gluten-free, and high-protein) presented difficulties in generating balanced and personalized meal plans. The system had to ensure that meals met all dietary restrictions while still providing adequate nutrition. This required extensive refinement of the decision-routing mechanism to ensure that each meal suggestion complied with all dietary constraints without compromising on nutritional quality.

Key Challenges in Dietary Restrictions:

I. Ensuring that meal suggestions were nutritionally balanced while adhering to multiple, overlapping dietary restrictions.

II. Developing prompt templates and decision-making systems that could handle a wide variety of complex dietary needs.

User Input Complexity

User input complexity was another significant challenge for Gini. Many users provided minimal or vague information, such as "I want healthy meals" or "Give me a balanced diet," which made it difficult for the system to generate accurate or personalized responses. The system struggled to interpret these general inputs, leading to a need for more robust **decision-routing mechanisms** that could guide the system to ask follow-up questions or prompt users for more specific details.

Key Solutions:

I. **Clarification Prompts**: Implemented mechanisms to request additional information from users when inputs were too vague, ensuring that the system could still generate meaningful responses.

II. **Refinement of Routing Logic**: The routing system was improved to handle ambiguous queries more efficiently by selecting the appropriate prompt chain and adjusting the input as necessary.

Conclusion

Lessons Learned from Using LLMs

Integrating **Large Language Models (LLMs)** like GPT-3.5 into Gini provided valuable insights into the strengths and limitations of natural language processing for dietary and nutritional systems. The LLM was highly effective in generating flexible, natural responses, allowing Gini to handle a range of queries related to meal planning, recipes, and nutrition. The ability of the model to understand and generate human-like text greatly enhanced user interaction, creating a conversational, user-friendly experience.

However, LLMs also presented challenges, particularly when handling **incomplete or ambiguous inputs**. Users often provided vague or underspecified queries (e.g., "I want something healthy"), which required implementing decision-routing mechanisms to guide the model toward generating meaningful responses. Without such routing and clarification strategies, the LLM tended to struggle with understanding the user's true intent. This experience underscored the importance of refining prompt templates and enhancing the system's ability to request additional information when needed.

I. LLMs are highly effective for creating natural, conversational interfaces but require additional mechanisms to handle vague or incomplete inputs.

II. Decision-routing and prompt refinement are critical for optimizing the accuracy and relevance of responses.

Overall Project Reflection

The development of Gini resulted in a successful **Minimum Viable Product (MVP)**, encompassing core features like personalized meal planning, recipe suggestions, and nutrient breakdown. The system achieved its primary goals of offering tailored meal recommendations based on user dietary preferences and providing clear nutritional data for various food items. Additionally, the multi-prompt chain system allowed Gini to effectively route user inputs to the appropriate model prompts, enhancing the overall accuracy of responses.

Despite these accomplishments, there are several areas where Gini could be expanded and improved. **User preference memory** was identified as a key feature that would allow the system to store and recall user data across sessions, providing a more personalized experience. Furthermore, integrating **real-time nutritional APIs** like USDA FoodData Central would significantly improve the accuracy and range of Gini's food data. These enhancements, along with expanding the system's handling of complex dietary restrictions, represent opportunities for future growth.

The project successfully developed a functional MVP with essential features, but future improvements could focus on personalization, API integration, and handling more complex dietary requirements.

Recommendations for Future Projects

Based on the development process and lessons learned from Gini, the following recommendations are proposed for future iterations or similar projects:

I. **Enhance Personalization**:

  a. Incorporating long-term **user data storage** would greatly enhance the personalization of the system. By allowing Gini to remember user preferences across sessions, the system could offer more consistent and tailored meal recommendations over time. This would also open the door to features like long-term meal tracking and progress monitoring.

II. **Broaden API Integration**:

b. Future projects should consider integrating additional external APIs, such as **USDA FoodData Central** or **Spoonacular**, to increase the accuracy and variety of nutritional data. Access to real-time food databases would ensure that users receive the most up-to-date nutritional information, allowing for more precise meal planning and nutrient breakdowns.

III. **Expand User Testing**:

c. Conducting broader **user testing** is essential for gathering diverse feedback and identifying areas for improvement. Testing with users from different dietary backgrounds (e.g., individuals with specific restrictions like keto, paleo, or FODMAP diets) would help refine the system's ability to handle complex dietary needs and ensure it remains relevant to a wider audience.

Key Recommendations:

I. Incorporate user preference memory to improve session-to-session personalization.

II. Integrate additional APIs to access real-time, accurate nutritional data.

III. Expand user testing to capture diverse dietary needs and preferences, improving system robustness and adaptability.

Justification & Self-Evaluation

Personally, I, Swapnil Jha, feel that the project was not just justified but also needed by me in general daily usage, I was looking for such a product to use so that I can eat healthier as I have been losing weight ever since I moved to Germany, this project module gave me the push needed to work on this project.

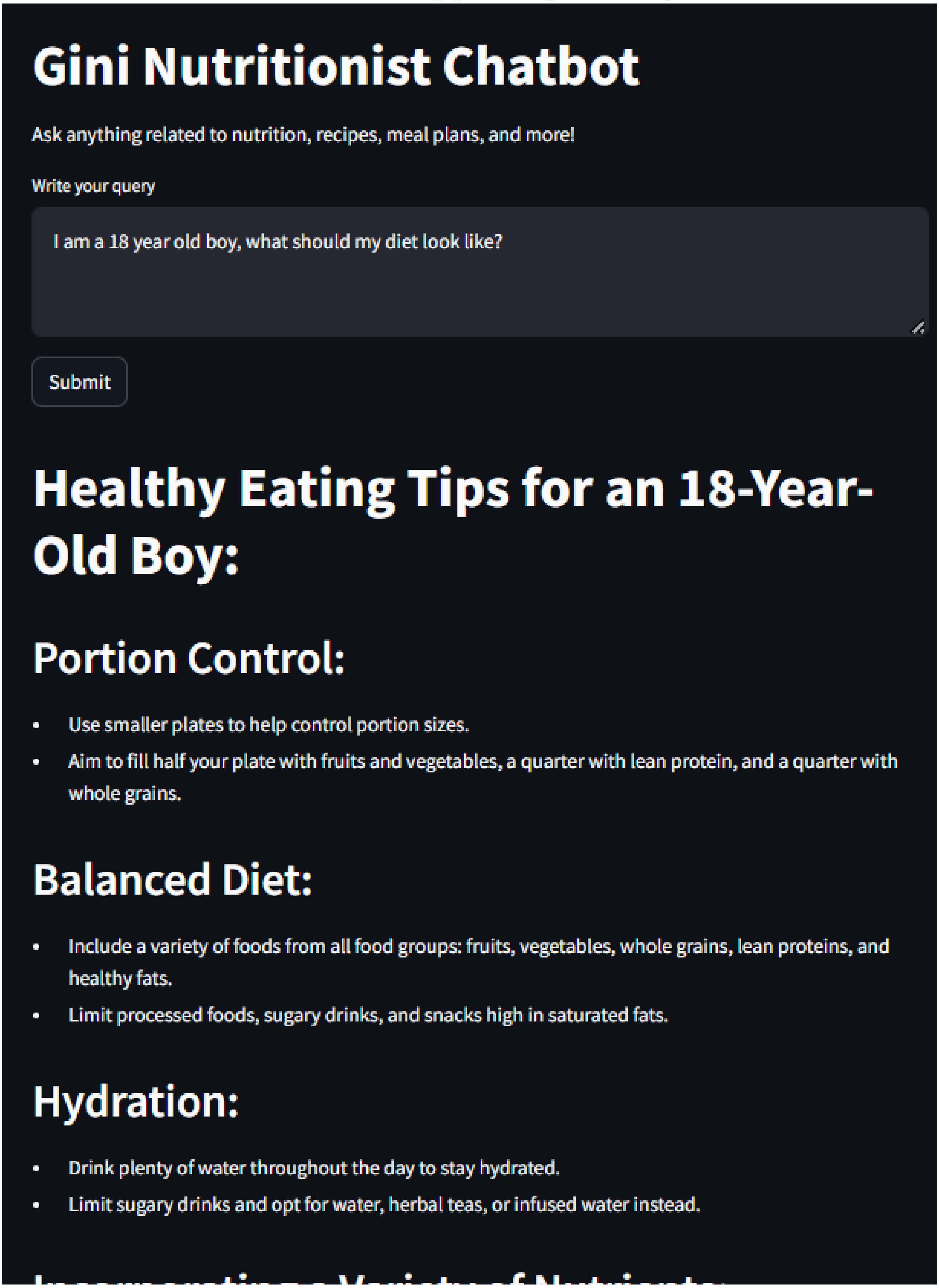
The project is hard to evaluate with a validation metric, as for all LLM purposes it would perform on the same level as GPT3.5, and to cross-verify the factual accuracy, it would be a manual effort to do so, and I and my team checked a lot of output manually, and as far as we could check, it was all correct. Products of these kinds should be checked by the community with constant feedback and correction and that's how this can be improved in the future. For now, it is fulfilling the purpose for which it was made.

In conclusion, I would say this project was justified, and the decision to go with a pre-trained model was good, in another timeline, where we would have a lot of money, fine-tuning such a model on a carefully curated dataset would be optimal, but for now, a pre-trained off the shelf model is fulfilling the requirements which I had from this project.

References

- https://medium.com/@ranjithkumar.panjabikesanind/build-a-decision-evaluation-ai-harnessing-the-power-of-mental-models-langchain-openai-and-llm-8082a9ec3a46
- https://www.geeksforgeeks.org/build-chatbot-webapp-with-langchain/
- https://medium.com/@dash.ps/build-chatbot-with-llms-and-langchain-9cf610a156ff
- https://arxiv.org/pdf/2403.00781
- https://smith.langchain.com/hub/
- https://medium.com/@sathiyamurthi239/how-to-create-a-streamlit-executable-python-to-exe-3bcb8eed9b16
- https://docs.streamlit.io/deploy/streamlit-community-cloud/deploy-your-app

GINI Web App Sample Usage



pLLM: Guidance on Nutritional Intake (GINI) Code Repository and Dependencies

Swapnil Jha (swapnil.jha@uni-potsdam.de)

Cognitive Systems at University of Potsdam

Code Repository –

- https://github.com/swapniljha001/GINI

Web App Link –

- https://nutrigini.streamlit.app/

Libraries Used –

- Streamlit (https://pypi.org/project/streamlit/)
- Langchain (https://pypi.org/project/langchain/)
- Langchain Community (https://pypi.org/project/langchain-community/)
- Langchain Core (https://pypi.org/project/langchain-core/)
- Langchain OpenAI (https://pypi.org/project/langchain-openai/)
- IPython (https://pypi.org/project/ipython/)

References

- https://medium.com/@ranjithkumar.panjabikesanind/build-a-decision-evaluation-ai-harnessing-the-power-of-mental-models-langchain-openai-and-llm-8082a9ec3a46
- https://www.geeksforgeeks.org/build-chatbot-webapp-with-langchain/
- https://medium.com/@dash.ps/build-chatbot-with-llms-and-langchain-9cf610a156ff
- https://arxiv.org/pdf/2403.00781
- https://smith.langchain.com/hub/
- https://medium.com/@sathiyamurthi239/how-to-create-a-streamlit-executable-python-to-exe-3bcb8eed9b16
- https://docs.streamlit.io/deploy/streamlit-community-cloud/deploy-your-app

pLLM: Guidance on Nutritional Intake (GINI) Personal Statement of Contribution

Swapnil Jha (swapnil.jha@uni-potsdam.de)

Cognitive Systems at University of Potsdam

Swapnil Jha's Statement of Contribution

In the group project to develop a LLM-powered nutrition chatbot, I, Swapnil Jha, led the backend development, integration, and testing efforts, ensuring the seamless functioning of Gini's prompt chains and StreamLit interactions. I was responsible for building the backend infrastructure that facilitated smooth communication between the chatbot and users, while integrating multiple components of the system to ensure reliability and scalability with the StreamLit Community Hosting System.

I also played a pivotal role in prompt engineering, where I designed and implemented a multi-prompt chain system using LangChain. This system dynamically routed user inputs to appropriate response mechanisms, allowing for more accurate and contextually relevant answers. The LangChain integration was crucial in enhancing the chatbot's ability to process varied nutritional queries and provide well-informed responses.

My contributions encompassed the technical backend, where I integrated OpenAI's GPT3.5 Turbo, StreamLit, and LangChain to create a working prototype on the system locale, and on the innovation side, where I adapted a multi-prompt architecture to improve the chatbot's user interaction and problem-solving capabilities.